



AgilePoint Developer User's Guide

AgilePoint BPMS v5.0 SP2

Document Revision r5.2.5

November 2011

Contents

Preface.....	4
Disclaimer of Warranty.....	4
Copyright.....	4
Trademarks.....	4
Government Rights Legend.....	4
Virus-free software policy.....	4
Document Revision Numbers.....	5
AgilePoint Documentation in PDF and HTML.....	5
Contacting AgilePoint Sales.....	5
Contacting Customer Support.....	6
AgilePoint Developer.....	7
Converting a Web Application to AgilePoint Format.....	8
Project Templates.....	9
AgilePart Project Template.....	9
AgileWork Project Template.....	9
AgileStub Project Template.....	9
AgileConnector Project Template.....	9
AgilePoint Web Application Project Template.....	10
Creating an AgilePoint Web Application.....	10
AgilePoint BPM Web Application Window.....	10
AgilePoint Web Controls.....	13
Adding AgilePoint Web Controls to Your Project.....	13
ConfirmButton.....	13
DatePicker.....	14
WFcheckbox.....	14
WFFileAttachment.....	15
WFRadioButton.....	15
WFDropDownList.....	16
WFGriDView.....	16
WFTaskGridControl.....	17
WFTextBox.....	19
WFProcessTemplateDropDownList.....	19
WFComment.....	20
WFRadioButtonList.....	20
WFListBox.....	21
WFUserNameDropDownList.....	21
WFRadioButtonPair.....	21
WFUserNameListBox.....	22
WFProcessViewer.....	22
Task List Web Part for AgilePoint Web Applications.....	23
Manual Setup.....	23
Settings.....	23
Linking to Tasks in Multiple Process Templates.....	23
Linking to Tasks in Multiple Web Applications.....	24

Setting the Maximum Number of Tasks to Display.....	24
Filter Tasks by Application Name.....	24
Show or Hide the Complete Selected Tasks Option.....	24
Limitations.....	24
Exception Handling for AgileParts.....	25
Developing the Custom Exception Handler AgileConnector.....	25
Deploying the Custom Exception Handler AgileConnector.....	25
Implementation.....	26
Testing and Debugging Custom AgileShape Projects.....	27
Ensuring You Are Testing the Correct Version.....	27
Process Merging and Splitting.....	28
Process Merging.....	28
Process Splitting.....	28
Remote API.....	29
Accessing AgilePoint Web Service API.....	29
Surrogating.....	29

Preface

Disclaimer of Warranty

AgilePoint, Inc. makes no representations or warranties, either express or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

Copyright

Copyright © 2011 AgilePoint, Inc. All rights reserved.

Trademarks

AgilePoint, Inc. and AgilePoint's products are trademarks of AgilePoint Inc. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

Government Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the applicable license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14, as applicable.

Virus-free software policy

AgilePoint recognizes that viruses are a significant security consideration for our customers. To date, we have had no report of AgilePoint BPMS carries any virus. AgilePoint takes the following measures to ensure our software is free of viruses upon delivery:

- AgilePoint is built on top of Microsoft .NET framework. The pre-compiled executable is a .NET Common Language Runtime (CLR) application, not a native machine binary. As far as is known at this time, there are no viruses that infect .NET CLR executables.
- The virtual environment for the product packaging process is fully isolated and protected, and anti-virus software is installed and running during packaging.
- The deliverable package is scanned by anti-virus software before upload to our customer download site.

Document Revision Numbers

AgilePoint documentation uses the revision number format **rX.Y.Z**. The letters and numbers in this revision number can be interpreted as follows:

- **r** - Indicates "revision." This helps to differentiate the document *version* numbers, which start with **v**.
- **X** - The major version number for AgilePoint BPMS to which this document refers. For example, AgilePoint releases 5.0, 5.0 SP1, and 5.5 would all have an **X** value of **5**.
- **Y** - The major document revision number. This number typically changes only when either there is a new AgilePoint release, or there are major changes to the document.
- **Z** - The minor document revision number. This number is incremented each time the document is republished.

AgilePoint Documentation in PDF and HTML

AgilePoint documentation is provided in both print-friendly (PDF) and web-based (HTML) formats.

Advantages of HTML Documentation

- HTML is the **primary delivery format** for AgilePoint documentation.
- Unified, global **search** across all documentation. PDF documents allow you to search only within the context of a given PDF file.
- **All hyperlinks supported**. Links in PDFs are only supported in certain contexts.
- "One-stop shopping" for all information related to AgilePoint BPMS.
- The HTML documentation is updated more frequently than the PDF documentation. Web-based documentation is updated periodically between AgilePoint releases to address errors and omissions, but the PDF documentation is updated only at the time of a software release.

Advantages of PDF Documentation

PDFs can be more easily **printed**, **archived**, and **transferred** (such as by FTP or email) than HTML documentation.

For more information, see [Downloading Files and Sharing Links from the Documentation Library](#) on the [AgilePoint Support Portal](#).

Contacting AgilePoint Sales

AgilePoint is a leading Business Process Management System (BPMS) provider created by a team of driven people who strive to incorporate the principles of relentless innovation for the benefit of our customers. Our mission is to help companies of any size attain and sustain operational success through process excellence.

Headquarters: AgilePoint Corporation 1916C Old Middlefield Way Mountain View, CA 94043, USA

Tel: (650) 968 - 6789

Fax: (650) 968 - 6785

Email: info@agilepoint.com

Web site: www.agilepoint.com

International: For AgilePoint EMEA and AgilePoint Asia Pacific, please call the AgilePoint Corporate Office for contact information.

Contacting Customer Support

To contact AgilePoint Support, please submit a ticket on the AgilePoint Support Portal: <http://support.agilepoint.com/SupportPortal/>

If you do not have a Support Portal account, you can send an email to request one: support@agilepoint.com

AgilePoint Developer

AgilePoint Developer is a component of AgilePoint BPMS used by software developers to create reusable modules and extensions to develop highly complex and customized workflow management solutions. In short, AgilePoint Developer enables you to create any process management customizations you want that aren't included with AgilePoint BPMS out of the box. It is a Microsoft Visual Studio.NET add-in.

This document is a user's guide and reference to the features, functionality, usage, configuration, and administration of the AgilePoint Developer component of the AgilePoint BPMS Suite.

The following languages are currently supported by the VB. NET project template: English, Japanese, Korean, Simplified Chinese, Traditional Chinese, and Hebrew. Other languages (such as Spanish) are not currently supported, but can be enabled with some manual work. If your language is not supported an error will be returned and you can contact the AgilePoint Customer Support team (support@aagilepoint.com) for further assistance.

Converting a Web Application to AgilePoint Format

To add additional Web form pages to the application, click Tools > Convert to AgilePoint Web Application.

Project Templates

This section provides a general overview of the features, functionality, purpose, and intended usage of the project templates included with the AgilePoint Developer component of the AgilePoint BPMS Suite.

These project templates can be accessed using the **File > New > Project** menu command.

AgilePart Project Template

This project template is used to create a new custom AgilePart. AgilePart assemblies must be deployed into the AgilePoint Server's GAC. Deploying the AgilePart assemblies can be done manually via drag and drop or by using the AgilePart Deployment Utility. (You can deploy to your local environment by right-clicking the project name in the Solution Explorer pane in Visual Studio.)

AgileWork Project Template

This project template is used to create a new custom AgileWork. AgileWork assemblies (like AgilePart assemblies) must be deployed into the AgilePoint Server's GAC. Deploying the AgileWork assemblies must be done manually via drag and drop or by using the .NET Framework tool gacutil.exe. (You can deploy to your local environment by right-clicking the project name in the Solution Explorer pane in Visual Studio.)

AgileStub Project Template

This project template is used to create a new AgileStub that will customize the run time behavior of a specific process model.

An AgileStub is connected and initiated when a new process instance is initiated. An AgileStub's start and end lifecycle is one to one with the process instance life-time.

AgileConnector Project Template

This project template is used to create a new custom AgileConnector. AgileConnector assemblies (like AgilePart assemblies) must be deployed into the AgilePoint Server's GAC. Deploying the assemblies must be done manually via drag and drop or by using the .NET Framework tool gacutil.exe. (You can deploy to your local environment by right-clicking the project name in the Solution Explorer pane in Visual Studio.)

AgilePoint Web Application Project Template

ASP.NET Web applications can be used in conjunction with AgilePoint as a foundation for workflow deployments.

This project template is used to create a new AgilePoint-enabled ASP.NET Web application that will implement the user interfaces of a specific process model based on the Generic process template from AgilePoint Envision.

If specified, it will also create mobile pages for the web application. This enables easy support for mobile devices.

Creating an AgilePoint Web Application

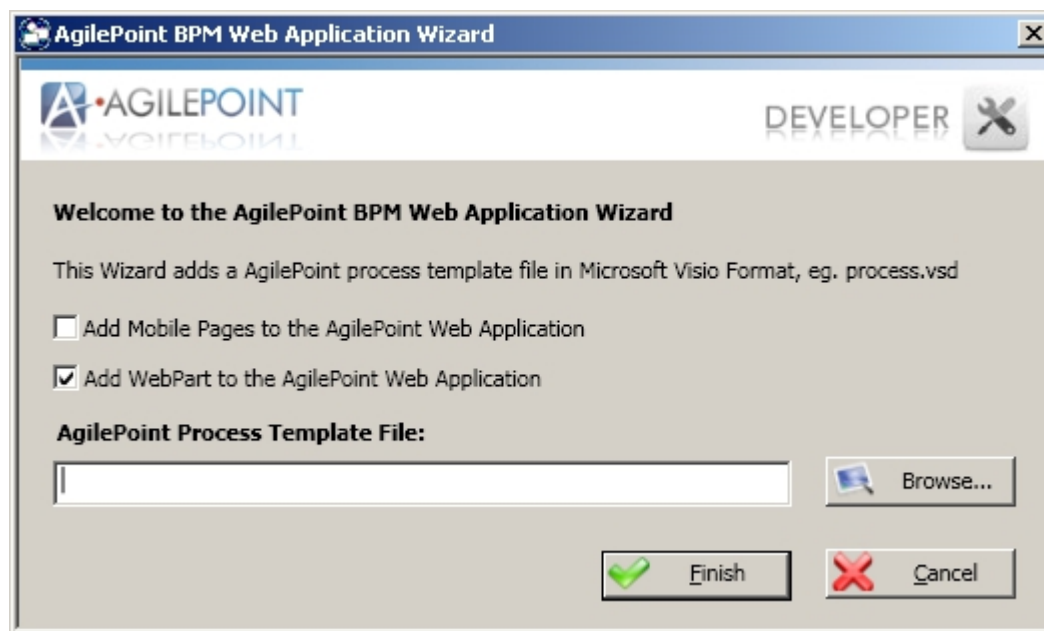
To create a new AgilePoint web application:

1. In AgilePoint Developer, click **File > New > Web Site**.
2. Specify the **Name** and **Location** for the web application, and click **OK**.
3. Complete the fields on the [AgilePoint BPM Web Application Wizard](#), and click **Finish**.


AgilePoint Developer creates the necessary skeleton in Visual Studio for the project.

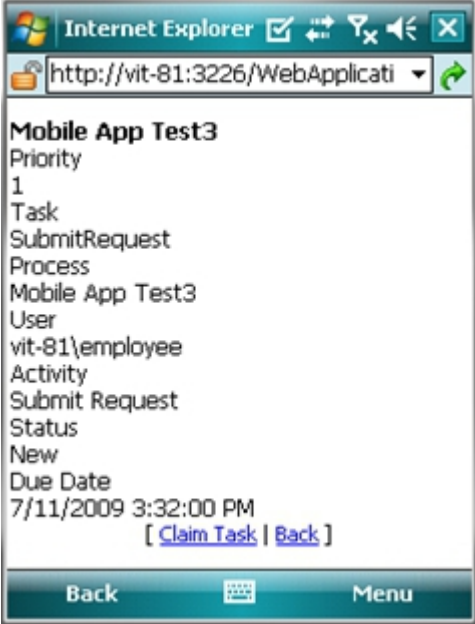
AgilePoint BPM Web Application Window

This window enables you to specify the options for a web application.



Field Definitions

Field Name	Definition
Add Mobile Pages to the AgilePoint Web Application	<p>If selected, AgilePoint Developer creates mobile versions of all files for the web application. This simplifies development for mobile devices.</p> <p>AgilePoint Developer adds the following mobile files:</p> <ul style="list-style-type: none"> • Mobile_LognonForm - A form page where the user can log on to the web application. • Mobile_Default - A default mobile template for an ASP.NET page. • Mobile_TasksPage - A mobile page that displays the user's Task List. There is also a detail page that displays details about a task when selected in the Task List. • A mobile page for the Work To Perform property of each activity that has the EnableMobile property set to True. <p>The following images show the mobile Task List and Task Detail pages:</p> 

Field Name	Definition
	
Add Web Part to the AgilePoint Web Application	<p>If selected, AgilePoint Developer creates the following Web Parts for the web application:</p> <ul style="list-style-type: none"> • Task List • Process Instance List
AgilePoint Process Template File	<p>Specifies the process model you want to associate with the web application.</p> <p>When you create the process template file, skeleton ASP.NET pages are created for each of the Work to Perform properties of the Manual Activities specified in the process model.</p>
Finish	Submits the options specified on the current window.

AgilePoint Web Controls

The following information provides details on how to leverage the built-in AgilePoint Web Controls to create process-enabled Web Applications.

The main difference between the AgilePoint Web Controls and the standard ASP.NET Web Controls is the additional AgilePoint specific properties that allow you to specify a binding name. With the AgilePoint Web Controls no code is needed to save and retrieve the values entered by the user. AgilePoint will automatically save the values to a custom attribute that can be accessed from anywhere in the process.

Adding AgilePoint Web Controls to Your Project

1. In Microsoft Visual Studio .NET, click the Toolbox button.
2. Right-click in the Toolbox and click Add Tab.
3. Name the new tab AgilePoint Web Controls (or your preferred name).
4. Right-click on the AgilePoint Web Controls tab and click Choose Items.
5. From the .NET Framework Components tab, deselect all the checked items and scroll down and select all the AgilePoint Workflow controls. The AgilePoint Workflow controls can be identified in the list with the Namespace Ascentn.Workflow.WebControls.
6. Click OK.
7. Expand the AgilePoint Web Controls tab. The list of AgilePoint Web Controls are displayed. You can now make use of these Web Controls to build your .aspx pages.

ConfirmButton

The implementation of this Web Control allows you to call your custom JavaScript function and post back to the server to execute managed code.

Properties


Inherits

- System.Web.UI.WebControls.Button

Usage

- Insert similar JavaScript as below in the .aspx file. The last line of the JavaScript function must include "return true" in order to run server-side post back. If not, after the confirmation message dialog is displayed, no action will be taken.

```
<script language="javascript">
  function ConfirmSubmission()
  {
    alert('Are you sure you want to submit?');
    return true;
  }
</script>
```



- The ConfirmButton Web Control does not support the ASP.NET RequiredFieldValidator control. This is a limitation of this AgilePoint Web Control, if you have RequiredFieldValidators setup for other input Web Controls on the page, then ConfirmButton will not support the validation action.

DatePicker

The implementation of this Web Control provides a convenient way for the user to pick a date.

Properties

Inherits

- System.Web.UI.Control
- INamingContainer
- WFDataBindingControl

Usage

Not applicable.

WFcheckbox

The implementation of this Web Control shows a check box on the page at run time.

Properties

Inherits

- System.Web.UI.WebControls.checkbox
- WFDataBindingControl

Usage

Not applicable.

WFFileAttachment

The implementation of this Web Control uploads a single file, either to the system directory or SharePoint during run time.

If the file already exists, information about the file is shown and the delete option is presented.

Properties

Inherits

- System.Web.UI.Design.ControlDesigner

Usage

- As a general rule do not use the & character in file names. If the uploading file name contains an "&" character, it will be changed to "_", because the "&" is the XML parser's primary key.
- The user who uploaded the file is the only user who can see the information in the file or delete the file.
- An empty file cannot be uploaded.

WFRadioButton

The implementation of this Web Control shows a radio button on the page at run time.

Properties

Inherits

- System.Web.UI.WebControls.RadioButton
- WFDataBindingControl

Usage

- Usually WFRadioButton is used to bind with Single Condition. The data type of WFRadioButton is Boolean only. Single Condition only supports Boolean data type.
- Only set the binding name to the radio button that represents 'yes' or 'true'. For the other radio button in the group (the one represents 'no' or 'false'), do not set the binding name/leave the binding name property empty. If you set the binding name for both radio buttons in the group, the auto-binding does not work. See ManagerApproval.aspx for example.
- WFRadioButton is not the same as WFRadioButtonList. WFRadioButtonList only supports string data type. Do not use WFRadioButtonList to bind with a Single Condition, rather use a Multiple Condition AgileShape.

WFDropDownList

The implementation of this Web Control shows a drop-down list for which to select a value on the page at run time.

Properties

Inherits

- System.Web.UI.WebControls.DropDownList
- WFDataBindingControl

Usage

Not applicable.

WFGridView

The implementation of this Web Control shows the associated XML Schema content in a repeating table view at run time. Add, update and delete is enabled.

Properties

Inherits

- System.Web.UI.WebControls.GridView
- WFDataBindingControl

Usage

After Setting the BindingName property, add the desired fields in accordance with the XML schema.

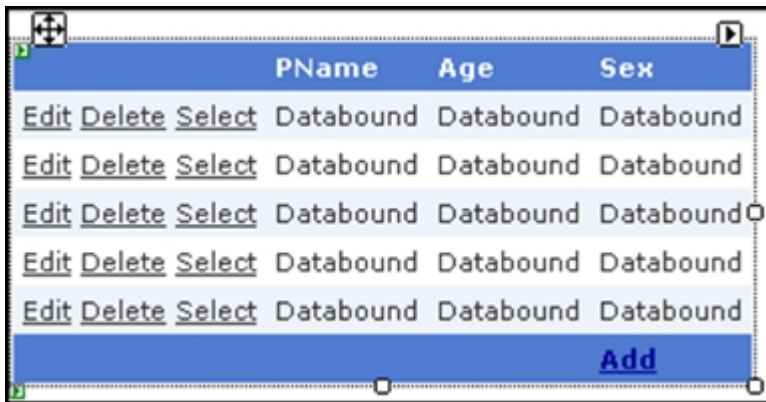
The structure of selected XML node must be similar to the following:

```
<Persons>
  <Person>
    <PName>Tom</PName>
    <Age>23</Age>
    <Sex>Male</Sex>
  </Person>
  <Person>
    <PName>Jack</PName>
    <Age>25</Age>
    <Sex>Male</Sex>
  </Person>
</Persons>
```

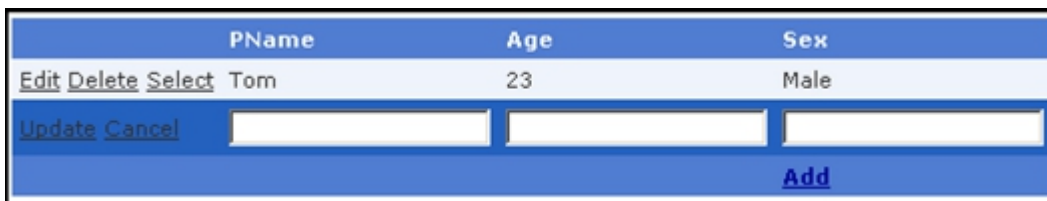
The value of AutoGenerateColumns property should not be true; otherwise it will generate an error.

Button can be added, updated or deleted by setting the CommandName. Without coding in the event of [RowUpdated](#) and [RowDeleted](#), WFGridView will save the changes to the datasource automatically.

Add a **Submit** button that calls `GetBoundDataItem()` to save the changes to XML.



At run time the Web Control shows the XML content.



Click the corresponding button to add, update, or delete the repeating section.

Click the **Submit** button, all changes will be saved to AgilePoint Server.

WFTaskGridControl

The implementation of this Web Control shows a user Task List on the page at run time. The AgilePoint Task List supports the following functions:

- Pool Function (not in column)
- Reassign Task
- Cancel Task
- Multiple Complete
- Multiple Cancel
- Rollback
- At design time, columns must be added manually, as a result the value of the `AutoGenerateColumns` property must be set to `False`, otherwise it will throw an error.

- At run time, you can view the tasks in the task grid view.
- You can do sorting and grouping by clicking the title.
- Upon selecting the check box for a task, the row will become bold.
- When right-clicking on a row, a menu appears with several functions.
- You can format the style of grid view as you want by setting the grid view's property or "auto Format" command.

Properties

Inherits

- System.Web.UI.WebControls.GridView

Usage

You must create a QueryTaskListData event by double-clicking the QueryTaskListData property.

Here is some sample code in the QueryTaskListData Event.

```
IWFWorkflowService api = GetAPI();
    string statusList = string.Format("{0};{1};{2}",
        WFManualWorkItem.ASSIGNED, WFManualWorkItem.OVERDUE,
        WFManualWorkItem.PSEUDO);
    WFManualWorkItem[] wks =
        api.GetWorkListByUserID(GetCurrentUser(), statusList);

    if (wks == null) wks = new WFManualWorkItem[0];
    TaskGridControl.QueryTaskListEventArgs qArgs =
        e as TaskGridControl.QueryTaskListEventArgs;
qArgs.Data = wks;
```

You can write TaskGridControl1.Binding(); in the update page function.

If you have set a command in the Context Menu, you need to design the RowCommand event by double-clicking the RowCommand property.

Below is sample code:

```
int index;
    if (!int.TryParse(e.CommandArgument as string,
        out index)) return;
    DataKey data = TaskGridControl1.DataKeys[index];
    string wid = data.Values["WorkItemID"] as string;
    string TaskForm = data.Values["Name"] as string;
    string pid = data.Values["ProcInstID"] as string;
    switch (e.CommandName)
    {
        case "DoTask":
            Response.Redirect("../" + TaskForm +
                ".aspx?WID=" + wid);
            break;
        case "ViewProcess":
            string url = "ProcessViewer.aspx?PIID=" + pid;
            string script = @"<script
```

```

        language='javascript'> " +
        @"var szFeatures = 'scrollbars=yes,
        width=770,height=500,resizable=yes';" +
        @"window.top.open('" + url + "',
        'ProcessViewer', szFeatures);" + @"
        </script> ";
        Page.ClientScript.RegisterStartupScript(Page.GetType(),
        "", script);
        break;
    case "CreateLinkWorkItem":
        IWFFrameworkService api = base.GetAPI();
        api.CreateLinkedWorkItem(wid, TaskForm,
        "Administrator", null, null);
        break;
    case "Test":
        int[] i = this.TaskGridControl1.GetSelectedIndices();
        IWFFrameworkService apil = base.GetAPI();
        foreach (int s in i)
        {
            DataKey d = TaskGridControl1.DataKeys[s];
            string id = data.Values["WorkItemID"] as string;
            //apil.CompleteWorkItem(id);
        }
        break;
    default:
        break;
}
UpdatePage();

```

WFTextBox

The implementation of this Web Control shows a text box for entering data on the page at run time.

Properties

Inherits

- System.Web.UI.WebControls.TextBox
- WFDataBindingControl

Usage

Not applicable.

WFProcessTemplateDropDownList

The implementation of this Web Control shows a drop-down list to allow the user to select a process template that is saved to a custom attribute or XML field that is defined in the Binding Name property.

Properties

Inherits

- WFDropDownList

Usage

Not applicable.

WFComment

The implementation of this Web Control records and displays comments on the page. It consists of two parts, one is a text box where the user can input comments, and the other is a display box where comment history is shown. The comment history box will be hidden if previous comments do not exist.

If the user wants to input special characters in the comment, the page attribute `ValidateRequest` should be set to **false**:

Properties

Inherits

- System.Web.UI.WebControls.CompositeControl
- WFDataBindingControl

Usage

Not applicable.

WFRadioButtonList

The implementation of this Web Control displays multiple radio buttons on the page.

Properties

Inherits

- System.Web.UI.WebControls.RadioButtonList
- WFDataBindingControl

Usage

WFRadioButtonList only supports string data type. Do not use WFRadioButtonList to bind with a Single Condition, rather use a Multiple Condition.

WFListBox

The implementation of this Web Control displays a list box for which to select a value on the page.

Properties

Inherits

- System.Web.UI.WebControls.ListBox
- WFDataBindingControl

Usage

Not applicable.

WFUserNameDropDownList

The implementation of this Web Control displays drop-down list for which to select a User Name on the page.

Properties

Inherits

- System.Web.UI.WebControls.DropDownList
- WFDataBindingControl

Usage

Not applicable.

WFRadioButtonPair

The implementation of this Web Control displays a pair of radio buttons (e.g. Approve/Reject).

Properties

Inherits

- System.Web.UI.WebControls.WebControl
- WFDataBindingControl

Usage

This Web Control is recommended to be used with a Single Condition AgileShape.

WFUserNameListBox

The implementation of this Web Control displays a list box that includes a list of user names for which the user can make a selection.

Properties

Inherits

- System.Web.UI.WebControls.ListBox
- WFDataBindingControl

Usage

Not applicable.

WFProcessViewer

The implementation of this Web Control shows the real-time status of a process instance on the page.

The control displays the process image and the status of activities in the AgilePoint process. A pop-up displays activity information for the current running activity.

Properties

Inherits

- System.Web.UI.WebControls.WebControl
- System.Web.UI.ICallbackEventHandler

Usage

- A single page can not contain more than one WFProcessViewer.
- It is required to drag the WFProcessViewer control into a .aspx page that inherits the Ascentn.Workflow.WebControls.WFWorksheetPage class.

Task List Web Part for AgilePoint Web Applications

The AgilePoint ASP.NET web application template provides a page with an ASP.NET version of the Task List Web Part.

Similar to the SharePoint Task List Web Part, the **Task** name opens a drop-down with a list of task management features.

Users can also reassign a selected task or cancel a selected task or process using the **Global actions** list.

Manual Setup

The following steps must be completed before running an ASP.NET web application with the Task List Web Part:

1. Set up the personalization database for the Web Part. Execute the command `aspnet_regsql` as follows:

```
"C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_regsql.exe"  
-E -S [hostname] -A all
```

Where `[localhost]` points to the default MS SQL Server instance running on the local machine.

Running this command creates a database with name **aspnetdb**. This can be changed to any value and the same should be reflected in the connection string in the next step. This step is a one-time execution as the personalization database will work for multiple Web Parts.

2. Open the `web.config` file and replace the connection string **LocalSqlServer** under the `<connectionStrings>` node with the location you defined in the previous step:

```
<add name="LocalSqlServer"  
connectionString="Data Source=[hostname];Initial  
Catalog=aspnetdb;trusted_connection=true"  
providerName="System.Data.SqlClient" />
```

Settings

You can modify the following settings.

Linking to Tasks in Multiple Process Templates

By default, the Task List Web Part displays the options **Open Task** and **Take Assignment and Open Task** for the process instance whose process template is attached with the AgilePoint Web Application. This is defined in the following setting in the `web.config`:

```
<add key="ProcessModel" value="LeaveRequest_ASP" />
```

If you want to add additional process templates, you can add the process template names to the **value** property of this node, separated by semicolons (;).

Linking to Tasks in Multiple Web Applications

You can also display the **Open Task** option for other web applications. Add the following **appSetting** node. The **key** is the name of the process template. The **value** is the base URL of the web application.

```
<add key="BudgetRequest" value="http://[hostname]:[port]/BudgetRequest" />
```

Setting the Maximum Number of Tasks to Display

You can modify the number of tasks to display on the Task List Web Part using the following **appSettings** tag entry:

```
<add key="PageSize" value="5" />
```

The default value is 5, but this number can be changed.

Filter Tasks by Application Name

Tasks can be filtered based on the application name using the following **appSettings** tag entry:

```
<add key="ExcludedApplication" value="SPSIntegration" />
```

This can accept multiple values with semicolon (;) separated application names.

Show or Hide the Complete Selected Tasks Option

You can show or hide the Complete Selected Tasks option in the Global Tasks menu by changing the value of the following **appSettings** node:

```
<add key="ShowCompleteSelectedTask" value="false" />
```

Limitations

The following limitations apply to the Task List Web Part:

- While grouping, the user cannot filter, and vice versa.
- The user cannot group the Select and Tasks columns.

Exception Handling for AgileParts

AgilePoint makes available to customers options for flexibility in exception handling for out-of-the-box, and custom AgileParts by enabling the ability to implement custom logic. For out-of-the-box exception handling functionality, AgilePoint writes AgilePart exception information (e.g. error message and status) to custom attributes to be viewed via AgilePoint Enterprise Manager or reported on later. The following information will show how to extend the out-of-the-box exception handling capabilities for AgileParts to call on a custom exception handler AgileConnector to perform more refined custom exception handling.

AgilePoint Envision includes a drop-down property called **ExceptionHandlerScope** for AgileParts that enables the user to select either **Local** or **Global**. If **Local** is selected, the exception handling will be handled as normal by reporting exception data to the **SaveErrorMessageTo** and **SaveStatusTo** custom attributes. If **Global** is selected, the AgilePart will continue to process as normal by reporting exception data to the **SaveErrorMessageTo** and **SaveStatusTo** custom attributes, but also includes the extended ability to call on a custom AgileConnector with custom exception handling logic to handle the exception(s) as desired. Now users have the extended ability to implement whatever logic is required to handle exceptions (e.g. custom error handling based on error message or error code, database interactivity, etc.).

Developing the Custom Exception Handler AgileConnector

1. Open AgilePoint Developer and create a new .NET Class Library project. Add the following 3 .NET References:
 - System.EnterpriseServices
 - System.Web
 - System.Web.Services
2. Add the following AgilePoint References:
 - Ascentn.Workflow.Share
 - Ascentn.Workflow.WFBase
3. Include the **Using Clause**: Ascentn.Workflow.Base
4. Implement the Base Class.
5. Implement the custom code.

For more information, see [AgileConnector Sample for Custom Exception Handling](#) on the [AgilePoint Support Portal](#).

Deploying the Custom Exception Handler AgileConnector

1. Copy the assembly (e.g. CustomExceptionHandler.dll) to the bin folder of AgilePoint Server.

2. Open the AgilePoint Server Configuration tool by selecting **Start > Programs > AgilePoint > AgilePoint Server Configuration**.
3. Click the **Extension** link.
4. Click **Add**. The **Global Extended Module** window appears.
5. In the **Name** field, type **CustomExceptionHandler**.
6. In the **Assembly** field, click the **Ellipses** button.
7. Select the **CustomExceptionHandler.dll** and click **Open**. The **Class Name** field is automatically populated.
8. Click **OK**. The AgilePoint Configuration window reappears with CustomExceptionHandler in the list of extensions.
9. If there are additional configuration properties associated with this AgileConnector, click **Configure**. The custom form window appears.
10. Configure the rules.
11. Click **OK**.

The configuration information is stored in the AgilePoint Server Configuration file.

Implementation

After development and deployment of the Global Exception Handler AgileConnector, the implementation is done at the process model layer via AgilePoint Envision. All out-of-the-box and custom AgileParts include a drop-down property activity called **ExceptionHandlerScope**. This drop-down is used to select either **Local** or **Global** where if **Local** is selected, the exception handling will be handled as normal by reporting exception data to the **SaveErrorMessageTo** and **SaveStatusTo** custom attributes and if **Global** is selected, the AgilePart will continue to process as normal by reporting exception data to the **SaveErrorMessageTo** and **SaveStatusTo** custom attributes, but also calls on a custom AgileConnector with custom exception handling logic to handle the exception(s) as desired.

Testing and Debugging Custom AgileShape Projects

Custom AgileShapes (including AgileParts and AgileWorks) are not stand-alone applications, and are only intended to be fully executed within the context of an AgilePoint Server instance. Like other "hosted" code (e.g. ASP.NET code-behind code; ASP.NET user and server controls; SharePoint modules; etc.), the behavior of the code at run time depends upon the state of the server. Although this makes unit testing more difficult than for a non-hosted application, it is an unfortunate consequence of the hosted-component model.

Therefore, at this time our official recommendation is to test AgileShapes using actual process models on a real AgilePoint Server installation, because this most closely emulates a production environment. This can be done either manually and/or using custom test applications that use the AgilePoint APIs to start and monitor the test processes.

However, there are certain steps you can take to make unit testing of your code more feasible. The first step is to separate as much of your AgileShape's code as possible into static methods and/or into separate class libraries. This should allow such separated code to be more easily unit testable outside of the hosting environment.

Another alternative is to implement a "fake server" by creating one or more custom classes that implement the AgilePoint interfaces required by your code at run time (e.g. IWFApi or IWFWorkflowService). The complexity of implementing such a solution will depend on how closely you need the "fake server" to mimic a real server. Although this would likely require more time than most developers would wish to spend, this would be the most comprehensive way to make AgilePoint-hosted components fully unit testable.

Ensuring You Are Testing the Correct Version

When testing and debugging custom AgileShapes, you must ensure that AgilePoint Envision is using the correct version:

1. When making minor implementation changes to an AgilePart or AgileWork, use the same assembly version number when you recompile. (I.e. make sure that the AssemblyInfo file in your project uses a hard-coded version number, and does not include any wildcards.)
2. After compiling a new version, add it to the GAC, or use the AgilePart Deployment Utility to deploy it.
3. If AgilePoint Server is already running and has already loaded an older copy of the assembly into memory, then you will need to reset IIS (e.g. run "iisreset") or restart the AgilePoint Server application in order to force AgilePoint Server to load the newer copy of the assembly into memory the next time it is needed.
4. If you compile a new copy of the assembly with a new assembly version number (as opposed to #1 above), then you will also need to unregister and reregister the AgileShape's assembly in Envision. However, if you followed the recommendations in #1 above, then you do not need to unregister or reregister the AgileShape's assembly from Envision (other than the single, initial registration of the assembly before you first use it).

Process Merging and Splitting

This section provides information about the Process Splitting and Merging functionality.

Download <http://download.agilepoint.com/pub/ProcessSplittingMergingSample.zip> for more information on implementing this functionality using the AgilePoint Web Service API. The AgilePoint Web Service API document also provides information about this functionality.

Process Merging

This feature can merge multiple process instances (at least two process instances) into one process instance, and the original instances will be canceled.

Process Splitting

This feature can split one process instance into multiple process instances (at least two process instances to begin with). The original process instance will still keep running.

The split process instances should be based on one process instance.

Remote API

This section provides information about the AgilePoint remote API. With AgilePoint Developer, AgilePoint can be developed and deployed as an embedded or autonomous application to meet different requirements.

- An embedded BPM application is usually tightly integrated with its surrounding system through AgilePoint API and functionality of an embedded BPMS can be exhibited through various surrounding applications.
- An autonomous AgilePoint Application is hosted by Microsoft IIS, running independently as a Microsoft .NET Web Application, and provides XML Web Service to its client Web Application. Exposing BPM functionality in this manner has a number of advantages. The greatest is that the Web Service can be provided to many applications independent of hardware, operating system, and programming languages boundaries. It makes application debugging easier because requests / responses are sent / received as human-readable content.

For more information, see [AgilePoint API](#) on the [AgilePoint Support Portal](#).

Accessing AgilePoint Web Service API

This section provides information about accessing the AgilePoint Remote API from a 3rd party application such as a SharePoint component or ASP.NET application in a separate machine through Impersonation. This method is used if you want to implement your own custom component to access AgilePoint through Web Service API. These components can be located in a physically separate machine (for example, calling from a SharePoint Web Part).

When accessing AgilePoint Server through the Remote API, an HTTP - 401 authentication error may occur. For the SharePoint component, it runs under the identity that is specified in the SharePoint Application Pool. In some case, the identity could be set using the NetworkService or LocalSystem. These identities will not be able to get authenticated remotely through IIS and therefore returned the HTTP - 401 error.

In a multiple-server environment, it is recommend to set up a dedicated user credential that the 3rd party application code can impersonate to get authenticated cross machine in order to access the AgilePoint Web Service. This dedicated user credential is typically stored in the web.config file of the application which can be retrieved before accessing the Web Service.

AgilePoint also provides an API called Surrogate() to allow the custom code to set the login user identity after the initial authentication through the impersonation. This will allow the remaining operation with the Web Service to run under the correct login identity.

Surrogating

1. In your custom coding for surrogating, at `api.SetClientAppName("MyApplication")`, put in your desired application name (case sensitive).
2. Open the **AgilePoint Server Configuration** and click the **Extension** link, then click the **Add** button.

3. Enter the application name (case sensitive) you used in the custom coding and also the Impersonator name (see picture below). The Impersonator has to be a registered user and preferably an administrator in AgilePoint Server.